

REMARKS

As a preliminary matter, applicant appreciates the indication of allowable subject matter in claim 10, which is being rewritten as independent claim 31. Allowance is requested.

As another preliminary matter, applicant appreciates the courtesy extended in an interview on February 9, 2006. As indicated in the Interview Summary mailed February 16, 2006, the examiner's interpretation of the '098 reference was discussed, without reaching agreement. The Examiner gave a broad interpretation of "wherein the number of threads...preserved and restored" as merely "limiting context switching". Applicant explained that the examiner's interpretation is unreasonably broad because it ignores the feature specified in the claim that "the number of threads in the second execution mode at any one time is limited". This is a concrete limitation in the claim language. Applicant respectfully submits that the broadest reasonable interpretation must take this feature into account. The examiner's interpretation reflects the effect of the feature as defined (in limiting the number of times the second processor context is preserved and restored) but not the feature itself.

Claim 2 has been cancelled, and claim 8 has been amended as required.

Claims 1-9, 11, 13-14, 27 and 30 stand rejected under § 103 on the basis of Zahir and Rosenthal. Applicant traverses this rejection because even combined, the references do not disclose or suggest limiting the number of threads in a second execution

mode at any one time, to limit the number of times that the second processor context is preserved and restored, as in amended claim 1.

As the examiner points out in the office communication mailed December 16, 2005, Zahir fails to specifically teach that “the number of threads in the second execution mode at any one time is limited, to limit the number of times that the second processor context is preserved and restored.” A single thread is a process that uses processor resources. The threads share these resources on a time-share basis.

The examiner cites Rosenthal and suggests that it would have been obvious to one of ordinary skill in the art at the time that the invention was made to combine Zahir with Rosenthal.

However, even with such a combination, a person of ordinary skill in the art would not have arrived at the feature that “the number of threads in the second execution mode at any one time is limited, to limit the number of times that the second processor context is preserved and restored.”

Rosenthal teaches a system and method for context switching of devices connected to a memory management unit (MMU). Devices connected through the MMU are controlled using the MMU page fault mechanism and page fault handler in each segment. Thus, addresses are allocated in the process address space for each process to provide for device addressing mapped into one segment of each process address space that will access the device. Furthermore, the “valid bits” associated with each page in a segment are turned on and off by the process in order to control the device. If the valid bits are off and an

attempt is made to access the device at the relevant address, a page fault occurs. The page fault mechanism invokes the page fault handler of the corresponding segment, and this is used to form a context switch of the device.

There is no mention in Rosenthal of a specific first execution mode with a first processor context, and a specific second execution mode with a second processor context. However, different devices require a different context (see column 2, lines 45-53).

As will be explained in the following, Rosenthal avoids unnecessary context switching. However, minimizing unnecessary context switching does not limit the number of threads in a particular execution mode at any one time which is larger than another execution mode.

The methods of context switching in Rosenthal determine which processes require context switching, and can control when the context switches are performed, in order to minimize the number of context switches (see column 4, lines 5-55). This is a known technique. For example, in the embodiment leading to the effect defined in column 7, line 4 to 7 and cited by the examiner, context switching of devices is controlled in that the valid bits are initially not set for the devices. This leads to a page fault when a process first attempts to access a device. There is a context switch when the process is activated and a context switch when the process is deactivated. The next time the process attempts to access the device when it has access to the CPU, a page fault occurs and a context switch is performed to restore the context on the device. There is no context switch if the process does not access the device during the CPU cycle in which it is active. This decreases the amount of context

switching simply in that context switching is only performed when it is required in that CPU cycle. There is no limiting of the number of threads or processes in a larger execution mode, as defined in amended claim 1.

In a further embodiment, a process uses a device and is therefore running in a large execution mode and is then transferred to a smaller context execution mode. It is possible to save the context required for the second device mode when the transfer is made to the first mode. However, according to this embodiment such a save is delayed until the next time a thread enters the second execution mode. If it is the same process or thread, no save will be necessary and the context switch will have been avoided. Column 7, lines 35 to 57 relate to this embodiment, in which a page fault occurs only when a thread other than the thread previously in the second execution mode attempts to access the device. This leads to the delaying mechanism mentioned above. Equally, it does not impose any upper limit on the number of threads in the larger execution mode.

Applicant has not found any other indication that the number of threads in the larger execution mode is limited in the other passages cited by the examiner, or indeed the entire document. The passages cited on column 11, lines 21 to 30 relate to prioritization. Effectively, if one process must execute before a second process, the valid bits are set so that even if the second process attempts to access the device first, a page fault occurs, and the process is halted until the first process has completed access.

In conclusion, Rosenthal teaches reduction of context switching by not switching context unless necessary, and delaying context switching as appropriate.

Rosenthal does not, however, teach any limitation whatsoever on the number of threads in a larger execution mode. There is a clear linguistic and technical distinction between limiting the number of threads in the second execution mode (to a certain number of threads) which involves counting these threads, comparing them to a maximum and taking action to ensure that the limit is not overstepped, and using mechanisms which may under certain circumstances act automatically to reduce the number of threads in this second execution mode. Limiting the number of threads has the advantage of giving certainty as to the amount of memory required, whereas reducing the number does not. Accordingly, withdrawal of the rejection of amended claim 1, and its related independent claims is requested. Withdrawal of the rejection of independent claims 27 and 30 is requested for the reasons given with respect to claim 1.

Claims 12, 15 and 19 stand rejected on the basis of various references. Applicant traverses these rejections for the reasons given with respect to independent claim 1. Withdrawal is requested.

For the foregoing reasons, applicants believe that this case is in condition for allowance, which is respectfully requested. The examiner should call applicant's attorney if an interview would expedite prosecution.

Respectfully submitted,

GREER, BURNS & CRAIN, LTD.

By 
Patrick G. Burns
Registration No. 29,367

March 13, 2006

300 South Wacker Drive
Suite 2500
Chicago, Illinois 60606
Telephone: 312.360.0080
Facsimile: 312.360.9315

Customer No. 24978